Coalition Formation towards Energy-Efficient Collaborative Mobile Computing

Liyao Xiang, Baochun Li, Bo Li
Aug. 3, 2015
Collaborative Mobile Computing

- **Mobile offloading**: migrating the computation-intensive portion of an app to the cloud to execute.

- **Gain**: trades the relatively low communication energy expense for high computation power consumption.

- **Loss**: suffers high network latency.

- New features such as *Continuity* made offloading tasks to nearby devices possible.
Coalition Formation of Mobile Users

- Previous works assume fully cooperative mobile users.

- We assume users are:
  - cooperative: collaborates under agreements.
  - individually rational: prefers coalition if it benefits.

- We study the problem of coalition formation among a group of mobile users targeting at the same job.
Coalition Formation of Mobile Users

- User case: crowdsourcing, content sharing, indoor localization, etc.

- Key questions:
  - Given a job partitioned into several tasks, how does a group of users form coalitions?
  - Within each coalition, how to distribute the tasks to each user?
System Model

- **A centralized approach**: an arbitrator profiles user’s info, organizes users into groups, and assigns tasks to each group.

- **A distributed scheme**: mobile users exchange profiles with users targeting at the same job. Based on the estimated energy cost, users decide to merge into one group or split up.

- A profile is generated by program static analysis tools.
System Model

Image Capturing 15M cycles

Features Extraction 50M cycles

Find Match 100M cycles

An example of mapping tasks to a set of devices

Task graph

Resource graph
Task Distribution

- Objective: minimizing the overall energy expense over all partitions of the resource graph with placement constraints.

- $B$ is the set of all partitions. $T$ represents one coalition. $C(T)$ is the sum of the energy expense on all mobile devices in coalition $T$.

$$\min_{\mathcal{P} \in B} \sum_{T \in \mathcal{P}} \min C(T).$$
Task Distribution

- To assign the binary variable $s_{i,n}$ representing task i is to be executed on device n.

- Placement constraints:

$$
\sum_{n \in T} s_{i,n} = 1, \forall i \in V, \quad (5)
$$

$$
\sum_{i \in V} s_{i,n} \geq 1, \forall n \in T, \quad (6)
$$

$$
S_{i,n}s_{j,m}e_{i,j} \leq l_{n,m}, \forall i \neq j, \forall n \neq m, n, m \in T \quad (7)
$$

$$
S_{i,n} \leq r_{i,n}, \forall i, \forall n \in T, \quad (8)
$$

$$
S_{i,n} \in \{0, 1\}, \forall n \in T, \forall i \in V. \quad (9)
$$
Coalition Formation

- The centralized approach is non-convex and NP-hard. How about going distributed?

- Collaboration among mobile users is modelled as a non-transferrable utility coalition game \((N, v)\) where \(N\) is the entire set of users, and \(v\) is the utility for the coalition which is defined as the negative energy cost.

- Partition:

  **Definition 1:** A *collection* is any family \(T = \{T_1, ..., T_l\}\) of mutually disjoint coalitions. If additionally \(\bigcup_{j=1}^{l} T_j = N\), the collection \(T\) is called a *partition* of \(N\).
Coalition Formation

- Comparison relation:

  **Definition 2**: Assume $A$ and $B$ are partitions of the same set $C$, a *comparison relation* $\triangleright$ is defined as, $A \triangleright B$ means that the way $A$ partitions $C$ is preferable to the way $B$ partitions $C$.

- Pareto order: the transformation of coalitions through Pareto order can only happen when it at least *strictly improves* the utility of one user, i.e., given two partitions $T$ and $T'$, with $\phi(T)$ representing the energy cost of $T$, the comparison relation is expressed as:

  $$T \triangleright T' \iff \forall n, \phi_n(T) \leq \phi_n(T') \text{ and } \exists m, \phi_m(T) < \phi_m(T')$$
Coalition Formation

- Two rules to transform coalitions:
  
  **Merge:** \( \{T_1, \ldots, T_k\} \cup P \rightarrow \{\bigcup_{j=1}^k T_j\} \cup P \), where \( \{\bigcup_{j=1}^k T_j\} \succ \{T_1, \ldots, T_k\} \).

  **Split:** \( \{\bigcup_{j=1}^k T_j\} \cup P \rightarrow \{T_1, \ldots, T_k\} \cup P \), where \( \{T_1, \ldots, T_k\} \succ \{\bigcup_{j=1}^k T_j\} \).

- Based on the above rules, we derive the algorithm:

---

**Algorithm 1** Collaborative Computing Game through Merge and Split

Input: Initial partition \( T = \{T_1, \ldots, T_l\} = N \)

Output: Final partition \( T_{final} \)

repeat
  \( T = \text{Merge}(T) \);
  \( T = \text{Split}(T) \);
until merge and split terminates.

\( T_{final} = T \).
Stability Analysis

- Definition: we consider a partition $T$ is stable if for any collection $C$ of the entire user set $N$ that

$$C[T] \succ C, \quad (14)$$

where

$$C[T] = \{T_1 \cap \bigcup \{C\}, ..., T_k \cap \bigcup \{C\}\} \setminus \{\emptyset\}.$$  

- We prove that the stability defined above implies contractually individual stability, i.e., a state that no player can benefit from moving its coalition to another without making others worse off.
Dc-Stable

- We proved our merge-and-split mechanism is stable if allowing users to transfer between coalitions by merge and split. The stable partition is called Dc-stable partition.

- If a Dc-stable partition $T$ exists, then $T$ is the unique outcome of every iteration of merge and split.
Performance Evaluation

- Setup
  - Computation cycles of each task is 20-100 M cycles.
  - Data transferred is 10-1000 KB on each link.
  - Energy consumption in data transmission is 20-200mJ/KB.
  - Computation energy cost is 40-60 mJ/M cycles.
Performance Evaluation

- Average Energy Cost

<table>
<thead>
<tr>
<th>Average Energy Cost per User over All Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-coop</td>
</tr>
<tr>
<td>Energy Cost(J)</td>
</tr>
</tbody>
</table>

(a) Average energy cost using merge and split.

(b) Average energy cost using centralized algorithm (intractable when the number of users is beyond 7).
Performance Evaluation

- Average coalition size.
Performance Evaluation

- Average proportion of computation and communication cost.
Performance Evaluation

- Emulation for a real-world app & running time comparison.
Conclusion

- We formulate the task assignment problem as a 0-1 integer programming problem and use heuristic method to solve it.

- We devise a distributed merge-and-split algorithm to allow collaborative and individually rational users to form coalitions.

- We reveal the conditions under which the scheme yields a stable partition.
Q & A.
Thank you.